Lesson 12

多线程

Multi-Threads

需求场景

案例 1

Bathroom.java

```java
public class Bathroom {
    public void serve(Person person) {
        String message;

        message = String.format("%s 进 厕所", person.getName());
        System.out.println(message);

        for (int i = 0; i < 100; i++) {
            message = String.format("%s 正在使用厕所 %d%%", person.getName(), i + 1);
            System.out.println(message);
        }
        person.release();

        message = String.format("%s 出 厕所", person.getName());
        System.out.println(message);
    }
}
```

Person.java

```java
public class Person {

    // Mark - Context

    private Bathroom bathroom;

    public Bathroom getBathroom() {
        return bathroom;
    }

    public void setBathroom(Bathroom bathroom) {
        this.bathroom = bathroom;
    }

    // Mark - Basic

    private String name;

    public Person(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    // Mark - Drink

    private final static int capacity = 10;
    private int cups = 0;

    private void drink(int cups) {
        this.cups += cups;
        String message = String.format("%s 喝了 %d 杯水，当前状态 %d / %d", name, cups, this.cups, capacity)
        System.out.println(message);
        if (this.cups >= capacity) {
            bathroom.serve(this);
        }
    }

    public void release() {
        cups = 0;
```

```
    }

    // Mark - Life

    public void run() {
        for (int i = 0; i < 100; i++) {
            int cups = new Random().nextInt(3) + 1;
            drink(cups);
        }
    }
}
```

Office.java

```
public class Office {
    public static void main(String[] args) {
        Bathroom bathroom = new Bathroom();

        Person p1 = new Person("小明");
        Person p2 = new Person("小红");

        p1.setBathroom(bathroom);
        p2.setBathroom(bathroom);

        p1.run();
        p2.run();

    }
}
```

现象：

启动 Office

程序按顺序执行，第二函数需要等待。

诉求：

两个函数同时执行

案例 1

BlockingHandler.java

```java
public class BlockingHandler implements Handler {
    @Override
    public void handle(Request request, Response response) {
        try {
            Thread.sleep(10_000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        response.setStatus(Response.ok);
        response.setData("long");
    }
}
```

Server.java

```java
public class Server {
    public static void main(String[] args) {
        SRRPServer srrpServer = new SRRPServer();
        srrpServer.registerHandler("wait", new BlockingHandler());
        srrpServer.start();
    }
}
```

Server.java

```java
public class Server {
    public static void main(String[] args) {
        SRRPServer srrpServer = new SRRPServer();
        srrpServer.registerHandler("wait", new BlockingHandler());
        srrpServer.start();
    }
}
```

Client.java

```java
public class Client {
    public static void main(String[] args) {
        long time1 = System.currentTimeMillis();
        Response response = new SRRPClient().send(new Request("wait", ""));
        long time2 = System.currentTimeMillis();
        System.out.println(time2 - time1);
    }
}
```

现象：

运行一个 Server，连续启动两个 Client

第二个Client所花时间几乎是第一个的两倍

服务器只能做一件事，第二个客户端需要等待

诉求：

同时处理两个客户端请求

案例 3

Client.java

```java
public class UIClient extends Application {

    private Label label;

    @Override
    public void start(Stage primaryStage) throws Exception {

        Pane pane = new Pane();
        Scene scene = new Scene(pane, 400, 100);

        // textField
        TextField textField = new TextField();
        textField.setLayoutX(10);
        textField.setLayoutY(10);
        textField.setPrefWidth(100);
        textField.setPrefHeight(30);
        pane.getChildren().add(textField);

        // button
        Button button = new Button("申请");
        button.setLayoutX(120);
        button.setLayoutY(10);
        button.setPrefWidth(80);
        button.setPrefHeight(30);
        pane.getChildren().add(button);

        // label
        label = new Label("hello");
        label.setLayoutX(210);
        label.setLayoutY(10);
        label.setPrefWidth(100);
        label.setPrefHeight(30);
        pane.getChildren().add(label);


        button.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                Response response = new SRRPClient().send(new Request("wait", ""));
                System.out.println(response.getData());

                label.setText(response.getData());

            }
        });

```

```
47          primaryStage.setScene(scene);
48          primaryStage.show();
49      }
50
51      public static void main(String[] args) {
52          Application.launch(args);
53      }
54  }
```

现象：

运行 Server，运行 UIClient

点击按钮后客户端出现卡顿

客户端只能做一件事，等服务器回复，就不能响应新的用户请求

诉求：

等待的同时，响应用户交互

并发

改进 案例 1

Person.java

```
1  * public class Person implements Runnable {
2      ...
3
4      public void run {
5          ...
6      }
7  }
```

Office.java

```java
public class Office {
    public static void main(String[] args) {
        Bathroom bathroom = new Bathroom();

        Person p1 = new Person("小明");
        Person p2 = new Person("小红");

        p1.setBathroom(bathroom);
        p2.setBathroom(bathroom);

        new Thread(p1).start();
        new Thread(p1).start();

    }
}
```

线程 与 进程

线程	Thread	喷水壶	执行机
进程	Process	草坪	代码

一个程序启动后是一个进程。

同时开多个程序，是多个进程。

一个进程里，运行代码的是线程。

一般程序只有一个线程。

但一个进程可以启动多个线程，就是多个线程同时执行程序。

启动多线程

```
new <? extends Thread>().start();
new Thread(<? object implement Runnable).start();
```

同时也不同时

宏观：

多个线程同时执行代码，同时浇水

计算机组成：

```
DISK + MEMORY + CPU
2 ->
3 -> CPU -> 5
+ ->
    shock
```

统一时间 一个 CPU 只会计算一个

时间片分配：

CPU 会 高速切换 允许某一个线程使用它

这段时间，就说 CPU 分配给线程 时间片

切换的速度太快，以至于人类无法识别。

事实：

无法预测 哪个线程会得到时间片

无法预测 一个线程得到时间片的长度是多少

改进 案例 2

改进 案例 3

反并发

多线程 与 内存结构

一个进程内的所有线程 会共享 进程的资源（比如 堆）

每个线程 也有自己的资源 （比如 栈）

案例 1 问题

同时执行

但希望有些资源 不要支持 并发

案例 1 改进

Bathroom.java

```java
public class Bathroom {
    public void serve(Person person) {
        synchronized (this) {
            String message;

            message = String.format("%s 进 厕所", person.getName());
            System.out.println(message);

            for (int i = 0; i < 100; i++) {
                message = String.format("%s 正在使用厕所 %d%%", person.getName(), i + 1);
                System.out.println(message);
            }
            person.release();

            message = String.format("%s 出 厕所", person.getName());
            System.out.println(message);
        }
    }
}
```

相关名词

同步	Synchronize	单限	一个个执行
异步	Asynchronize	并发	一起执行

锁 与 钥匙

打开单限封锁门（加锁）：

看 synchronized 后面的值是多少，就需要那个地址作为钥匙进入这个门

如果 钥匙就在门上，那么进去并且拿走钥匙

但 钥匙也可能不在门上，那么就需要等待

退出单限封锁区（解锁）：

程序走出封锁区后，就需要归还当时的那把钥匙

此时 在门口等待 钥匙的人，由操作系统决定把钥匙给哪个线程。